
PPPPPPPPPPPP AAAAAAAAA TTTTTTTTTTTTTTT CCCCCCCCCCC HHH HHH
PPPPPPPPPPPP AAAAAAAAA TTTTTTTTTTTTTTT CCCCCCCCCCC HHH HHH
PPPPPPPPPPPP AAAAAAAAA TTTTTTTTTTTTTTT CCCCCCCCCCC HHH HHH
PPP PPP AAA AAA TTT CCC HHH HHH
PPP PPP AAA AAA TTT CCC HHH HHH
PPP PPP AAA AAA TTT CCC HHH HHH
PPP PPP AAA AAA TTT CCC HHH HHH
PPP PPP AAA AAA TTT CCC HHH HHH
PPPPPPPPPPPP AAA AAA TTT CCC HHHHHHHHHHHHHHH
PPPPPPPPPPPP AAA AAA TTT CCC HHHHHHHHHHHHHHH
PPPPPPPPPPPP AAA AAA TTT CCC HHHHHHHHHHHHHHH
PPP AAAAAAAAAAAAAAA TTT CCC HHH HHH
PPP AAAAAAAAAAAAAAA TTT CCC HHH HHH
PPP AAAAAAAAAAAAAAA TTT CCC HHH HHH
PPP AAA AAA TTT CCC HHH HHH
PPP AAA AAA TTT CCC HHH HHH
PPP AAA AAA TTT CCC HHH HHH
PPP AAA AAA TTT CCCCCCCCCCC HHH HHH
PPP AAA AAA TTT CCCCCCCCCCC HHH HHH
PPP AAA AAA TTT CCCCCCCCCCC HHH HHH

```
PPPPPPPP      AAAAAA    TTTTTTTTTT    SSSSSSSS  PPPPPPPP      AAAAAA
PPPPPPPP      AAAAAA    TTTTTTTTTT    SSSSSSSS  PPPPPPPP      AAAAAA
PP      PP   AA    AA       TT      SS          PP      PP   AA    AA
PP      PP   AA    AA       TT      SS          PP      PP   AA    AA
PP      PP   AA    AA       TT      SS          PP      PP   AA    AA
PP      PP   AA    AA       TT      SS          PP      PP   AA    AA
PPPPPPPP     AA    AA       TT        SSSSSS    PPPPPPPP     AA    AA
PPPPPPPP     AA    AA       TT        SSSSSS    PPPPPPPP     AA    AA
PP           AAAAAAAAAA     TT              SS  PP           AAAAAAAAAA
PP           AAAAAAAAAA     TT              SS  PP           AAAAAAAAAA
PP           AA    AA       TT              SS  PP           AA    AA      ....
PP           AA    AA       TT              SS  PP           AA    AA      ....
PP           AA    AA       TT      SSSSSSSS    PP           AA    AA      ....
PP           AA    AA       TT      SSSSSSSS    PP           AA    AA      ....

LL             IIIIII      SSSSSSSS
LL             IIIIII      SSSSSSSS
LL               II      SS
LL               II      SS
LL               II      SS
LL               II      SS
LL               II        SSSSSS
LL               II        SSSSSS
LL               II              SS
LL               II              SS
LL               II              SS
LL               II              SS
LLLLLLLLLL     IIIIII    SSSSSSSS
LLLLLLLLLL     IIIIII    SSSSSSSS
```

```
;     1          L 0001  0 MODULE PATSPA (%IF %VARIANT EQL 1
;     2                0002  0                %THEN
;     3                0003  0                        ADDRESSING_MODE (EXTERNAL = LONG_RELATIVE,
;     4                0004  0                                         NONEXTERNAL = LONG_RELATIVE),
;     5                0005  0                %FI
;     6                0006  0                IDENT = 'V04-000'
;     7                0007  0                ) =
;     8                0008  1 BEGIN
;     9                0009  1
;    10                0010  1 !
;    11                0011  1 !*****************************************************************************
;    12                0012  1 !*                                                                           *
;    13                0013  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;    14                0014  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;    15                0015  1 !*  ALL RIGHTS RESERVED.                                                     *
;    16                0016  1 !*                                                                           *
;    17                0017  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
;    18                0018  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
;    19                0019  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
;    20                0020  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
;    21                0021  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
;    22                0022  1 !*  TRANSFERRED.                                                             *
;    23                0023  1 !*                                                                           *
;    24                0024  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
;    25                0025  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
;    26                0026  1 !*  CORPORATION.                                                             *
;    27                0027  1 !*                                                                           *
;    28                0028  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
;    29                0029  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
;    30                0030  1 !*                                                                           *
;    31                0031  1 !*                                                                           *
;    32                0032  1 !*****************************************************************************
;    33                0033  1 !
;    34                0034  1
;    35                0035  1 !++
;    36                0036  1 ! FACILITY:     PATCH
;    37                0037  1 !
;    38                0038  1 ! ABSTRACT:     THIS ROUTINE HANDLES FREE PATCH AREA, ALIGNMENT, ALLOCATION, AND EXPANSION.
;    39                0039  1 !
;    40                0040  1 ! ENVIRONMENT:  VAX/VMS
;    41                0041  1 !
;    42                0042  1 ! AUTHOR: K.D. MORSE     , CREATION DATE: 17-NOV-77
;    43                0043  1 !
;    44                0044  1 ! MODIFIED BY:
;    45                0045  1 !
;    46                0046  1 !       V03-003 MTR0025         Mike Rhodes             11-Aug-1983
;    47                0047  1 !               Modify routine PAT$EXP_AREA to signal an ERROR (severity)
;    48                0048  1 !               message when an expansion request is made while patching
;    49                0049  1 !               a file in ABSOLUTE mode.  This will cause the current
;    50                0050  1 !               command to be aborted and the user is returned back to the
;    51                0051  1 !               PATCH command prompt.  Files may NOT be expanded in absolute
;    52                0052  1 !               mode, as could result from a command like:
;    53                0053  1 !                       PATCH>  REPLACE/INST 20='movl r0,r1'
;    54                0054  1 !                       NEW>    'movl r0,r1'
;    55                0055  1 !                       NEW>    'bneq 200'
;    56                0056  1 !                       NEW>    EXIT
;    57                0057  1 !
```

```
;    58          0058  1 !       V03-002 MTR0016         Mike Rhodes             03-Nov-1982
;    59          0059  1 !               Modify PAT$BUILD_ISE to accept one additional argument which
;    60          0060  1 !               is the address to be modified.  This address is used for INSERT
;    61          0061  1 !               and REPLACE commands when patching protected shareable images.
;    62          0062  1 !               The attributes of the image section which contains the address
;    63          0063  1 !               being modified will be propagated to the newly created default
;    64          0064  1 !               patch area.
;    65          0065  1 !
;    66          0066  1 !       V03-001 MTR0007         Mike Rhodes             14-Jun-1982
;    67          0067  1 !               Use shared system messages.  Affected modules include:
;    68          0068  1 !               DYNMEM.B32, PATBAS.B32, PATCMD.B32, PATIHD.B32, PATINT.B32,
;    69          0069  1 !               PATIO.B32, PATMAI.B32, PATMSG.MSG, PATWRT.B32, and PATSPA.B32.
;    70          0070  1 !
;    71          0071  1 !               The shared messages are defined by DYNMEM.B32's invocation of
;    72          0072  1 !               SHRMSG.REQ and we simply link against these symbols.  They are
;    73          0073  1 !               declared as external literals below.
;    74          0074  1 !
;    75          0075  1 !       V03-000 MTR0001         Mike Rhodes             15-Mar-1982
;    76          0076  1 !               Modify routine PAT$EXP_AREA to allow PIC SHR images to be
;    77          0077  1 !               patched using default patch area which may be expanded as
;    78          0078  1 !               needed.  Also, removed the old 50% growth area logic which
;    79          0079  1 !               has been made obsolete by the above change.
;    80          0080  1 !
;    81          0081  1 !       V02-008 MTR0001         Mike Rhodes             15-Sep-1981
;    82          0082  1 !               Modify routine PAT$BUILD_ISE.  The location algorithm
;    83          0083  1 !               for placing the PATCH ISE/ISD pair in the ISE list is
;    84          0084  1 !               as follows:
;    85          0085  1 !                       The PATCH ISE/ISD pair are located in the ISE list
;    86          0086  1 !                       FOLLOWING the last "Normal" ISD and PRECEDING the
;    87          0087  1 !                       first Non-Based Global or Stack ISDs.
;    88          0088  1 !
;    89          0089  1 !               Included in the modification is the definition of two new
;    90          0090  1 !               variables,  PREV_ISE_PTR - Pointer to Previous ISE,  and
;    91          0091  1 !                           TEMP         - Holds the FLINK from the previous
;    92          0092  1 !                                          ISE till its put into the new ISE.
;    93          0093  1 !
;    94          0094  1 !       V02-007 PCG0001         Peter George            02-FEB-1981
;    95          0095  1 !               Add require statement for LIB$:PATDEF.REQ
;    96          0096  1 !
;    97          0097  1 !       V0206   CNH0038         Chris Hume              4-Oct-1980      16:00
;    98          0098  1 !               Last Cluster will now remain set when new Patch Area is added.
;    99          0099  1 !               Patch Area will be allocated at a distance one half the size of
;   100          0100  1 !               the Last Cluster (beyond its end).
;   101          0101  1 !
;   102          0102  1 !       V0105   CNH0023         Chris Hume              16-Nov-1979     14:00
;   103          0103  1 !               Turn off ISD$V_LASTCLU for all ISD's when PATCH Area is added
;   104          0104  1 !               to an image.  Also unrecognized languages will now be processed
;   105          0105  1 !               as though they were MACRO.  (PATBLD.B32 V0117, PATMAI.B32 V0228)
;   106          0106  1 !
;   107          0107  1 !       V0104   CNH0015         Chris Hume              27-Sep-1979     11:30
;   108          0108  1 !               Changed GBLWARN message from a warning to an informational.
;   109          0109  1 !               Added section name to the signal.  Added EXPSHRPAT error.
;   110          0110  1 !               (PATMAI.B32 V0225, PATMSG.MDL V0203, PATARI.B32 V0112)
;   111          0111  1 !
;   112          0112  1 ! MODIFICATIONS:
;   113          0113  1 !
;   114          0114  1 !  NO   DATE            PROGRAMMER      PURPOSE
```

```
;  115          0115  1 !  --   ----          ----------      -------
;  116          0116  1 !
;  117          0117  1 !  01   07-MAR-78      K.D. MORSE      ADD ROUTINES PAT$ADD_PAL.
;  118          0118  1 !  02   25-APR-78      K.D. MORSE      CONVERT TO NATIVE COMPILER.
;  119          0119  1 !  03   13-JUN-78      K.D. MORSE      ADD FAO COUNTS TO SIGNALS.
;  120          0120  1 !
;  121          0121  1 !--
```

```
;  123     0122  1 !
;  124     0123  1 ! TABLE OF CONTENTS:
;  125     0124  1 !
;  126     0125  1
;  127     0126  1 FORWARD ROUTINE
;  128     0127  1         PAT$ALIGN_CMD : NOVALUE,                        ! Executes align command
;  129     0128  1         PAT$BUILD_ISE : NOVALUE,                        ! Builds an image section descriptor
;  130     0129  1         PAT$EXP_AREA : NOVALUE,                         ! Expands patch area
;  131     0130  1         PAT$ADD_PAL : NOVALUE;                          ! Adds entry to PAL
;  132     0131  1
;  133     0132  1 !
;  134     0133  1 ! INCLUDE FILES:
;  135     0134  1 !
;  136     0135  1 LIBRARY 'SYS$LIBRARY:LIB.L32';                          ! System structure definitions
;  137     0136  1 REQUIRE 'SRC$:PATPCT.REQ';                              ! Defines PSECTs
;  138     0176  1 REQUIRE 'SRC$:PATGEN.REQ';                              ! Defines context bits
;  139     0398  1 REQUIRE 'SRC$:VXSMAC.REQ';                              ! Defines common macros
;  140     0463  1 REQUIRE 'SRC$:PREFIX.REQ';                              ! Defines structure macros
;  141     0651  1 REQUIRE 'SRC$:PATPRE.REQ';                              ! Defines PATCH structures
;  142     0814  1 REQUIRE 'LIB$:PATDEF.REQ';                      ! Defines literals
;  143     0868  1 REQUIRE 'LIB$:PATMSG.REQ';                              ! Defines error message codes
;  144     1042  1 REQUIRE 'SRC$:BSTRUC.REQ';                              ! Defines basic structures
;  145     1118  1 REQUIRE 'SRC$:LISTEL.REQ';                              ! Defines list structures
;  146     1160  1 REQUIRE 'SRC$:DLLNAM.REQ';                              ! Defines symbol table entry offsets
;  147     1218  1 REQUIRE 'SRC$:SYSSER.REQ';                              ! Defines FAO output macros
```

```
;  R1250  1     SWITCHES LIST (SOURCE);
;  R1251  1
;  R1252  1     EXTERNAL ROUTINE
;  R1253  1              PAT$fao_out;            ! formats a line and outputs to the terminal
;  R1254  1
```

```
;  148     1300  1 !
;  149     1301  1 ! MACROS:
;  150     1302  1 !
;  151     1303  1
;  152     1304  1 !
;  153     1305  1 ! EQUATED SYMBOLS:
;  154     1306  1 !
;  155     1307  1
;  156     1308  1 !
;  157     1309  1 ! OWN STORAGE:
;  158     1310  1 !
;  159     1311  1 OWN
;  160     1312  1         PAT_AREA_NAME : VECTOR[4,BYTE] INITIAL(%ASCIC 'PAA'),    ! Next patch area name
;  161     1313  1         PA_NAME_DSC : VECTOR[2,LONG] INITIAL(A_LONGWORD-A_BYTE, CH$PTR(PAT_AREA_NAME, 1)); ! String descript
;  162     1314  1
;  163     1315  1 !
;  164     1316  1 ! EXTERNAL REFERENCES:
;  165     1317  1 !
;  166     1318  1 EXTERNAL
;  167     1319  1         PAT$GL_PAL_LHD : REF BLOCK[,BYTE],                       ! Patch area listhead
;  168     1320  1         PAT$GL_ERRCODE,                                          ! Error code
;  169     1321  1         PAT$GL_CONTEXT : BITVECTOR,                              ! Context bits
;  170     1322  1         PAT$GL_FLAGS : BITVECTOR [32],                           ! CLI flags.
;  171     1323  1         PAT$GL_IMGHDR : REF BLOCK[,BYTE],                        ! Pointer to image header
;  172     1324  1         PAT$GL_PATAREA : REF BLOCK[,BYTE],                       ! Free patch area descriptor pointer
;  173     1325  1         PAT$GL_IHPPTR : REF BLOCK[,BYTE],                        ! Pointer to patch area of image header
;  174     1326  1         PAT$GL_ISELHD,                                           ! ISE List Head
;  175     1327  1         PAT$GL_ISETAIL : REF BLOCK[,BYTE],                       ! Pointer to tail of ISE table
;  176     1328  1         PAT$GL_NEWVPNMX,                                         ! Max VPN of image sections in new image
;  177     1329  1         PAT$GL_NEWVBNMX,                                         ! Max VBN of image sections in new image
;  178     1330  1         PAT$GL_IMGBLKS,                                          ! Number of blocks in new image
;  179     1331  1         PAT$GL_ISVADDR : VECTOR[,LONG],                          ! Addresses of last image section mapped
;  180     1332  1         PAT$GL_HEAD_LST,                                         ! Head of command argument list
;  181     1333  1         PAT$GL_SYMTBPTR,                                         ! Pointer to current default symbol table
;  182     1334  1         PAT$GL_SYMHEAD;                                          ! Pointer to listhead entry for user-defined
;  183     1335  1
;  184     1336  1 EXTERNAL ROUTINE
;  185     1337  1         PAT$ALLOBLK : NOVALUE,                                   ! Allocates free storage
;  186     1338  1         PAT$CREMAP : NOVALUE,                                    ! Creates and maps image sections
;  187     1339  1         PAT$DEFINE_SYM : NOVALUE,                                ! Defines a symbol
;  188     1340  1         PAT$FIND_SYM,                                            ! Find symbol definition
;  189     1341  1         PAT$FREEZ,                                               ! Allocates and zeros free storage
;  190     1342  1         PAT$MAP_ADDR : NOVALUE;                                  ! Maps an image address
;  191     1343  1
;  192     1344  1 EXTERNAL LITERAL
;  193     1345  1 !
;  194     1346  1 ! Define shared message references.  (resolved @ link time)
;  195     1347  1 !
;  196     1348  1         PAT$_CLOSEIN,                                            ! Error closing input file.
;  197     1349  1         PAT$_CLOSEOUT,                                           ! Error closing output file.
;  198     1350  1         PAT$_OPENIN,                                             ! Error opening input file.
;  199     1351  1         PAT$_OPENOUT,                                            ! Error opening output file.
;  200     1352  1         PAT$_READERR,                                            ! Error reading from file.
;  201     1353  1         PAT$_SYSERROR,                                           ! System Service error.
;  202     1354  1         PAT$_WRITEERR;                                           ! Error writing to file.
;  203     1355  1
```

```
;  205     1356  1 GLOBAL ROUTINE PAT$ALIGN_CMD  : NOVALUE =                    ! Performs align commands
;  206     1357  1
;  207     1358  1 !++
;  208     1359  1 ! FUNCTIONAL DESCRIPTION:
;  209     1360  1 !
;  210     1361  1 !       This routine aligns a free patch area to the requested boundary,
;  211     1362  1 !       word, longword, quadword, or page.  The patch area bytes between the
;  212     1363  1 !       old address and the rounded address are lost for patching purposes.
;  213     1364  1 !       The symbol name provided in the command is entered into the symbol list
;  214     1365  1 !       with a value of the patch area address.  If the free patch area is not
;  215     1366  1 !       large enough to be rounded to the appropriate boundary, an error is
;  216     1367  1 !       SIGNALed and the alignment does not take place.  The free area
;  217     1368  1 !       descriptor remains unchanged.
;  218     1369  1 !
;  219     1370  1 !       If the symbol name was previously defined, a message is produced and
;  220     1371  1 !       the name is redefined to the new patch area address.
;  221     1372  1 !
;  222     1373  1 !       Aligning the patch area to a byte boundary will merely cause the
;  223     1374  1 !       symbol to be defined as the next free byte of patch area.
;  224     1375  1 !
;  225     1376  1 ! FORMAL PARAMETERS:
;  226     1377  1 !
;  227     1378  1 !       none
;  228     1379  1 !
;  229     1380  1 ! IMPLICIT INPUTS:
;  230     1381  1 !
;  231     1382  1 !       The symbol name descriptor is set up by the parser.
;  232     1383  1 !       The context bits have already been set up for the command.
;  233     1384  1 !       The user-defined symbol table has been initialized as has the
;  234     1385  1 !       free memory handler.
;  235     1386  1 !
;  236     1387  1 ! IMPLICIT OUTPUTS:
;  237     1388  1 !
;  238     1389  1 !       none
;  239     1390  1 !
;  240     1391  1 ! ROUTINE VALUE:
;  241     1392  1 !
;  242     1393  1 !       none
;  243     1394  1 !
;  244     1395  1 ! COMPLETION CODES:
;  245     1396  1 !
;  246     1397  1 !       none
;  247     1398  1 !
;  248     1399  1 ! SIDE EFFECTS:
;  249     1400  1 !
;  250     1401  1 !       The default patch area is aligned to the appropriate boundary.
;  251     1402  1 !       If there is not enough patch area to align, a new patch area is
;  252     1403  1 !       created.
;  253     1404  1 !
;  254     1405  1 !--
;  255     1406  1
;  256     1407  2 BEGIN
;  257     1408  2
;  258     1409  2 LITERAL
;  259     1410  2         ONE_BLOCK = 1;                                  ! Number of blocks to expand patch area by
;  260     1411  2
;  261     1412  2 LOCAL
```

```
;  262     1413  2          TEMP_SYMTB,                                     ! Temporary symbol table pointer
;  263     1414  2          ALIGN_FACTOR,                                   ! Alignment boundary
;  264     1415  2          DESC_PTR : REF BLOCK[,BYTE],                    ! String descriptor pointer
;  265     1416  2          SYM_ENTRY_PTR,                                  ! Pointer to symbol entry
;  266     1417  2          PATCH_AREA_ADR,                                 ! Address of aligned patch area
;  267     1418  2          PATCH_AREA_SIZ;                                 ! Size of aligned patch area
;  268     1419  2
;  269     1420  2  !++
;  270     1421  2  ! Output current patch area statistics before alignment.
;  271     1422  2  !--
;  272     1423  2  $FAO_TT_OUT('old patch area size:     !XL', .PAT$GL_PATAREA[DSC$W_LENGTH]);
;  273     1424  2  $FAO_TT_OUT('old patch area address:  !XL', .PAT$GL_PATAREA[DSC$A_POINTER]);
;  274     1425  2
;  275     1426  2  !++
;  276     1427  2  ! Check for conflicting patch area requests and set up alignment factor.
;  277     1428  2  ! The alignment factor is set to the number of bytes in a longword, word,
;  278     1429  2  ! byte, page, or quadword.
;  279     1430  2  IF .PAT$GL_CONTEXT[ALIGN_BYTE]
;  280     1431  2  THEN
;  281     1432  2          ALIGN_FACTOR = A_BYTE;
;  282     1433  2  IF .PAT$GL_CONTEXT[ALIGN_WORD]
;  283     1434  2  THEN
;  284     1435  2          ALIGN_FACTOR = A_WORD;
;  285     1436  2  IF .PAT$GL_CONTEXT[ALIGN_LONG]
;  286     1437  2  THEN
;  287     1438  2          ALIGN_FACTOR = A_LONGWORD;
;  288     1439  2  IF .PAT$GL_CONTEXT[ALIGN_QUAD]
;  289     1440  2  THEN
;  290     1441  2          ALIGN_FACTOR = A_QUADWORD;
;  291     1442  2  IF .PAT$GL_CONTEXT[ALIGN_PAGE]
;  292     1443  2  THEN
;  293     1444  2          ALIGN_FACTOR = A_PAGE;
;  294     1445  2
;  295     1446  2  !++
;  296     1447  2  ! Now round up image header patch area address and alter patch area
;  297     1448  2  ! size to reflect any lost bytes.
;  298     1449  2  !--
;  299     1450  2  PATCH_AREA_ADR = ((.PAT$GL_PATAREA[DSC$A_POINTER] + (.ALIGN_FACTOR-1))/.ALIGN_FACTOR) * .ALIGN_FACTOR;
;  300     1451  3  IF (.PATCH_AREA_ADR NEQA .PAT$GL_PATAREA[DSC$A_POINTER])      ! If rounding actually occurred
;  301     1452  3          OR (.PAT$GL_PATAREA[DSC$W_LENGTH] EQL 0)              ! or no patch space exists
;  302     1453  2  THEN
;  303     1454  3          BEGIN
;  304     1455  3          PATCH_AREA_SIZ = .PAT$GL_PATAREA[DSC$W_LENGTH] +
;  305     1456  3                           .PAT$GL_PATAREA[DSC$A_POINTER] - .PATCH_AREA_ADR;
;  306     1457  4          IF (.PATCH_AREA_SIZ LEQ 0)                          ! Check no patch area left
;  307     1458  3          THEN
;  308     1459  4                  BEGIN
;  309     1460  5                  IF (.PAT$GL_PATAREA[DSC$A_POINTER] EQLA .PAT$GL_IHPPTR[IHP$L_RW_PATADR])
;  310     1461  4                  THEN
;  311     1462  4                          PAT$EXP_AREA (ONE_BLOCK)            ! Get another block
;  312     1463  4                  ELSE
;  313     1464  4                          SIGNAL(PAT$_NOPATAREA, 2, .PAT$GL_PATAREA[DSC$A_POINTER],
;  314     1465  4                                  .PAT$GL_PATAREA[DSC$W_LENGTH]);
;  315     1466  6                  PATCH_AREA_ADR = ((.PAT$GL_PATAREA[DSC$A_POINTER] +
;  316     1467  4                          (.ALIGN_FACTOR-1))/.ALIGN_FACTOR) * .ALIGN_FACTOR;
;  317     1468  4                  PATCH_AREA_SIZ = .PAT$GL_PATAREA[DSC$W_LENGTH] +
;  318     1469  4                                   .PAT$GL_PATAREA[DSC$A_POINTER] - .PATCH_AREA_ADR;
```

```
;  319     1470  3                    END;
;  320     1471  3            PAT$GL_PATAREA[DSC$A_POINTER] = .PATCH_AREA_ADR;         ! Set rounded address in header
;  321     1472  3            PAT$GL_PATAREA[DSC$W_LENGTH] = .PATCH_AREA_SIZ;         ! Set rounded size in header
;  322     1473  2            END;
;  323     1474  2
;  324     1475  2    !++
;  325     1476  2    ! Output current patch area after alignment.
;  326     1477  2    !--
;  327     1478  2    $FAO_TT_OUT('new patch area size:     !XL', .PAT$GL_PATAREA[DSC$W_LENGTH]);
;  328     1479  2    $FAO_TT_OUT('new patch area address:  !XL', .PAT$GL_PATAREA[DSC$A_POINTER]);
;  329     1480  2
;  330     1481  2    !++
;  331     1482  2    ! Now enter the symbol into the user-defined symbol table with a value equal
;  332     1483  2    ! to the aligned patch area address.
;  333     1484  2    !--
;  334     1485  2    !SYM_ENTRY_PTR = PAT$FIND_SYM(.LIST_ELEM_EXP1(.PAT$GL_HEAD_LST)); ! Check for previously defined symbol
;  335     1486  2    !IF .SYM_ENTRY_PTR NEQA 0                                       ! Yes, was previously defined
;  336     1487  2    !THEN                                                           ! Output informational message
;  337     1488  2    !       BEGIN
;  338     1489  2    !       SIGNAL(PAT$_REDEFSYM, 4, .SYM_CHCOUNT(.SYM_ENTRY_PTR), SYM_NAME(.SYM_ENTRY_PTR),
;  339     1490  2    !              .SYM_VALUE(.SYM_ENTRY_PTR), .PATCH_AREA_ADR);
;  340     1491  2    !       SYM_VALUE(.SYM_ENTRY_PTR) = .PATCH_AREA_ADR;            ! Set new value
;  341     1492  2    !       END
;  342     1493  2    !ELSE
;  343     1494  2    TEMP_SYMTB = .PAT$GL_SYMTBPTR;
;  344     1495  2    PAT$GL_SYMTBPTR = .PAT$GL_SYMHEAD;
;  345     1496  2    PAT$DEFINE_SYM(.LIST_ELEM_EXP1(.PAT$GL_HEAD_LST), .PATCH_AREA_ADR, TRUE); ! Enter into list
;  346     1497  2    PAT$GL_SYMTBPTR = .TEMP_SYMTB;
;  347     1498  2
;  348     1499  2
;  349     1500  2    RETURN;
;  350     1501  1    END;                                                            ! End of PAT$ALIGN_CMD


                                                                .TITLE  PATSPA
                                                                .IDENT  \V04-000\

                                                                .PSECT  _PAT$PLIT,NOWRT,NOEXE,0

                                                        1C  00000 P.AAA:    .BYTE   28
20  61  65  72  61  20  68  63  74  61  70  20  64  6C  6F  00001           .ASCII  \old patch area size:     !XL\
        4C  58  21  20  20  20  20  20  3A  65  7A  69  73  00010
                                                        1C  0001D P.AAB:    .BYTE   28
20  61  65  72  61  20  68  63  74  61  70  20  64  6C  6F  0001E           .ASCII  \old patch area address:  !XL\
        4C  58  21  20  20  3A  73  73  65  72  64  64  61  0002D
                                                        1C  0003A P.AAC:    .BYTE   28
20  61  65  72  61  20  68  63  74  61  70  20  77  65  6E  0003B           .ASCII  \new patch area size:     !XL\
        4C  58  21  20  20  20  20  20  3A  65  7A  69  73  0004A
                                                        1C  00057 P.AAD:    .BYTE   28
20  61  65  72  61  20  68  63  74  61  70  20  77  65  6E  00058           .ASCII  \new patch area address:  !XL\
        4C  58  21  20  20  3A  73  73  65  72  64  64  61  00067

                                                                .PSECT  _PAT$OWN,NOEXE,2

                                                41  41  50  03  00000 PAT_AREA_NAME:
                                                                .ASCII  <3>\PAA\
                                                    00000003    00004 PA_NAME_DSC:
```

```
                                        .LONG   3
00000000' 00008                         .ADDRESS PAT_AREA_NAME+1                                                        ;

                                ISE$C_SIZE==            20
                                TXT$C_SIZE==            4
                                PAL$C_SIZE==            16
                                ASD$C_SIZE==            9
                                FWR$C_SIZE==            24
                                        .EXTRN  PAT$FAO_OUT, PAT$GL_PAL_LHD
                                        .EXTRN  PAT$GL_ERRCODE, PAT$GL_CONTEXT
                                        .EXTRN  PAT$GL_FLAGS, PAT$GL_IMGHDR
                                        .EXTRN  PAT$GL_PATAREA, PAT$GL_IHPPTR
                                        .EXTRN  PAT$GL_ISELHD, PAT$GL_ISETAIL
                                        .EXTRN  PAT$GL_NEWVPNMX
                                        .EXTRN  PAT$GL_NEWVBNMX
                                        .EXTRN  PAT$GL_IMGBLKS, PAT$GL_ISVADDR
                                        .EXTRN  PAT$GL_HEAD_LST
                                        .EXTRN  PAT$GL_SYMTBPTR
                                        .EXTRN  PAT$GL_SYMHEAD, PAT$ALLOBLK
                                        .EXTRN  PAT$CREMAP, PAT$DEFINE_SYM
                                        .EXTRN  PAT$FIND_SYM, PAT$FREEZ
                                        .EXTRN  PAT$MAP_ADDR, PAT$_CLOSEIN
                                        .EXTRN  PAT$_CLOSEOUT, PAT$_OPENIN
                                        .EXTRN  PAT$_OPENOUT, PAT$_READERR
                                        .EXTRN  PAT$_SYSERROR, PAT$_WRITEERR
                                        .WEAK   ACCESS_CHECK

                                        .PSECT  _PAT$CODE,NOWRT,2

                        07FC 00000      .ENTRY  PAT$ALIGN_CMD, Save R2,R3,R4,R5,R6,R7,R8,-                              ; 1356
                                                R9,R10
                5A 00000000G    EF  9E 00002    MOVAB   PAT$GL_SYMTBPTR, R10
                59 00000000G    EF  9E 00009    MOVAB   PAT$FAO_OUT, R9
                58 00000000'    EF  9E 00010    MOVAB   P.AAA, R8
                57 00000000G    EF  9E 00017    MOVAB   PAT$GL_CONTEXT, R7
                56 00000000G    EF  9E 0001E    MOVAB   PAT$GL_PATAREA, R6
                7E      00      B6  3C 00025    MOVZWL  @PAT$GL_PATAREA, -(SP)                                          ; 1423
                                58  DD 00029    PUSHL   R8
                69              02  FB 0002B    CALLS   #2, PAT$FAO_OUT
                50              66  D0 0002E    MOVL    PAT$GL_PATAREA, R0                                              ; 1424
                          04    A0  DD 00031    PUSHL   4(R0)
                          1D    A8  9F 00034    PUSHAB  P.AAB
                69              02  FB 00037    CALLS   #2, PAT$FAO_OUT
          03    67              06  E1 0003A    BBC     #6, PAT$GL_CONTEXT, 1$                                          ; 1430
                53              01  D0 0003E    MOVL    #1, ALIGN_FACTOR                                                ; 1432
          03    67              04  E1 00041 1$: BBC     #4, PAT$GL_CONTEXT, 2$                                          ; 1433
                53              02  D0 00045    MOVL    #2, ALIGN_FACTOR                                                ; 1435
          03    67              02  E1 00048 2$: BBC     #2, PAT$GL_CONTEXT, 3$                                          ; 1436
                53              04  D0 0004C    MOVL    #4, ALIGN_FACTOR                                                ; 1438
          03    67              03  E1 0004F 3$: BBC     #3, PAT$GL_CONTEXT, 4$                                          ; 1439
                53              08  D0 00053    MOVL    #8, ALIGN_FACTOR                                                ; 1441
          05    67              05  E1 00056 4$: BBC     #5, PAT$GL_CONTEXT, 5$                                          ; 1442
                53      0200    8F  3C 0005A    MOVZWL  #512, ALIGN_FACTOR                                              ; 1444
                52              66  D0 0005F 5$: MOVL    PAT$GL_PATAREA, R2                                              ; 1450
                50        04    A2  D0 00062    MOVL    4(R2), R0
                51        FF  A340  9E 00066    MOVAB   -1(ALIGN_FACTOR)[R0], R1
                51              53  C6 0006B    DIVL2   ALIGN_FACTOR, R1
```

```
                                53  C5 0006E        MULL3   ALIGN_FACTOR, R1, PATCH_AREA_ADR
                                55  D1 00072        CMPL    PATCH_AREA_ADR, R0                              ;  1451
                                04  12 00075        BNEQ    6$
                                62  B5 00077        TSTW    (R2)                                            ;  1452
                                5D  12 00079        BNEQ    10$
                                62  3C 0007B 6$:    MOVZWL  (R2), R2                                        ;  1455
                            52  50  C1 0007E        ADDL3   R0, R2, R1
                            51  52  C3 00082        SUBL3   PATCH_AREA_ADR, R1, PATCH_AREA_SIZ              ;  1456
                            54  51  46 14 00086     BGTR    9$                                              ;  1457
                        51  00000000G   EF  D0 00088    MOVL    PAT$GL_IHPPTR, R1                           ;  1460
                    14  A1              50  D1 0008F    CMPL    R0, 20(R1)
                                        0B  12 00093    BNEQ    7$
                                        01  DD 00095    PUSHL   #1                                          ;  1462
                00000000V   EF          01  FB 00097    CALLS   #1, PAT$EXP_AREA
                                        11  11 0009E    BRB     8$
                                        05  BB 000A0 7$:  PUSHR   #^M<R0,R2>                                 ;  1464
                                        02  DD 000A2    PUSHL   #2
                            006D811A    8F  DD 000A4    PUSHL   #7176474
                00000000G   00          04  FB 000AA    CALLS   #4, LIB$SIGNAL
                            50          66  D0 000B1 8$:  MOVL    PAT$GL_PATAREA, R0                      ;  1466
            51              53  04      A0  C1 000B4    ADDL3   4(R0), ALIGN_FACTOR, R1                     ;  1467
                                        51  D7 000B9    DECL    R1                                          ;  1466
                            51          53  C6 000BB    DIVL2   ALIGN_FACTOR, R1                            ;  1467
            55              51          53  C5 000BE    MULL3   ALIGN_FACTOR, R1, PATCH_AREA_ADR
                            51          60  3C 000C2    MOVZWL  (R0), R1                                    ;  1469
            50              51  04      A0  C1 000C5    ADDL3   4(R0), R1, R0
            54              50          55  C3 000CA    SUBL3   PATCH_AREA_ADR, R0, PATCH_AREA_SIZ
                            50          66  D0 000CE 9$:  MOVL    PAT$GL_PATAREA, R0                      ;  1471
                    04      A0          55  D0 000D1    MOVL    PATCH_AREA_ADR, 4(R0)
                            60          54  B0 000D5    MOVW    PATCH_AREA_SIZ, (R0)                        ;  1472
                            7E      00  B6  3C 000D8 10$:  MOVZWL  @PAT$GL_PATAREA, -(SP)                  ;  1478
                                    3A  A8  9F 000DC    PUSHAB  P.AAC
                            69          02  FB 000DF    CALLS   #2, PAT$FAO_OUT
                            50          66  D0 000E2    MOVL    PAT$GL_PATAREA, R0                          ;  1479
                                    04  A0  DD 000E5    PUSHL   4(R0)
                                    57  A8  9F 000E8    PUSHAB  P.AAD
                            69          02  FB 000EB    CALLS   #2, PAT$FAO_OUT
                            53          6A  D0 000EE    MOVL    PAT$GL_SYMTBPTR, TEMP_SYMTB                 ;  1494
                            6A  00000000G   EF  D0 000F1    MOVL    PAT$GL_SYMHEAD, PAT$GL_SYMTBPTR         ;  1495
                                        01  DD 000F8    PUSHL   #1                                          ;  1496
                                        55  DD 000FA    PUSHL   PATCH_AREA_ADR
                            50  00000000G   EF  D0 000FC    MOVL    PAT$GL_HEAD_LST, R0
                                    04  A0  DD 00103    PUSHL   4(R0)
                    00000000G   EF          03  FB 00106    CALLS   #3, PAT$DEFINE_SYM
                            6A          53  D0 0010D    MOVL    TEMP_SYMTB, PAT$GL_SYMTBPTR                 ;  1497
                                        04  00110      RET                                                 ;  1501

; Routine Size:  273 bytes,    Routine Base:  _PAT$CODE + 0000
```

```
;  352     1502  1 GLOBAL ROUTINE PAT$BUILD_ISE (ISE_PTR,VPN,VBN,PAGE_CNT,ADR) : NOVALUE = ! Builds an ISD and enters it into I
;  353     1503  1
;  354     1504  1 !++
;  355     1505  1 ! FUNCTIONAL DESCRIPTION:
;  356     1506  1 !
;  357     1507  1 !       This routine builds a new image section descriptor.  It is a normal
;  358     1508  1 !       type image section with read-write, copy-on-reference attributes.
;  359     1509  1 !       The virtual page number, virtual block number, and the page count
;  360     1510  1 !       are input parameters.  The address of the image section table entry,
;  361     1511  1 !       built around the image section descriptor, is returned.  The image
;  362     1512  1 !       section entry is linked into the table.
;  363     1513  1 !
;  364     1514  1 ! FORMAL PARAMETERS:
;  365     1515  1 !
;  366     1516  1 !       ISE_PTR - Pointer to image section entry built
;  367     1517  1 !       VPN - Virtual page number of image section
;  368     1518  1 !       VBN - Virtual block number of image section
;  369     1519  1 !       PAGE_CNT - Number of pages in image section
;  370     1520  1 !       ADR = [OPTIONAL] Address which is to be modified by the patch.
;  371     1521  1 !
;  372     1522  1 ! IMPLICIT INPUTS:
;  373     1523  1 !
;  374     1524  1 !       The image section table is set up.
;  375     1525  1 !
;  376     1526  1 ! IMPLICIT OUTPUTS:
;  377     1527  1 !
;  378     1528  1 !       A new image section descriptor is built.
;  379     1529  1 !
;  380     1530  1 ! ROUTINE VALUE:
;  381     1531  1 !
;  382     1532  1 !       none
;  383     1533  1 !
;  384     1534  1 ! COMPLETION CODES:
;  385     1535  1 !
;  386     1536  1 !       none
;  387     1537  1 !
;  388     1538  1 ! SIDE EFFECTS:
;  389     1539  1 !
;  390     1540  1 !       If the ADR parameter is included in the call, we will propagate the
;  391     1541  1 !       the image section attributes (of the image section containing the
;  392     1542  1 !       address specified by ADR) to the newly created default patch area.
;  393     1543  1 !
;  394     1544  1 !--
;  395     1545  1
;  396     1546  2 BEGIN
;  397     1547  2
;  398     1548  2 BUILTIN
;  399     1549  2         NULLPARAMETER;
;  400     1550  2
;  401     1551  2 LOCAL
;  402     1552  2         PFC : BYTE,                                     ! Page Fault Cluster size
;  403     1553  2         TYPE : BYTE,                                    ! Type of image section
;  404     1554  2         FLAGS,                                          ! Image section Flags
;  405     1555  2         IDENT,                                          ! Image section Ident
;  406     1556  2         PREV_ISE_PTR : REF BLOCK[,BYTE],                ! Pointer to previous Image Section table en
;  407     1557  2         TEMP : REF BLOCK[,BYTE],                        ! Holds the FLINK from previous ISE
;  408     1558  2         LOCAL_ISE_PTR : REF BLOCK[,BYTE],               ! Image section table entry pointer
```

```
;  409     1559  2          ISD_PTR : REF BLOCK[,BYTE];                                 ! Image section descriptor pointer
;  410     1560  2
;  411     1561  2      !++
;  412     1562  2      ! Allocate space for new image section table entry.
;  413     1563  2      ! ***** UNTIL SYSTEM IS UPDATED TO CONTAIN AN IDENT PERFORM TEST ON WHAT
;  414     1564  2      ! ***** SIZE TO USE.
;  415     1565  2      !--
;  416     1566  2      IF PAT$K_LENPRIV GTR ISD$K_LENPRIV
;  417     1567  2      THEN
;  418     1568  2          PAT$ALLOBLK(ISE$C_SIZE+PAT$K_LENPRIV, .ISE_PTR)
;  419     1569  2      ELSE
;  420     1570  2          PAT$ALLOBLK(ISE$C_SIZE+ISD$K_LENPRIV, .ISE_PTR);
;  421     1571  2
;  422     1572  2      !++
;  423     1573  2      ! Now link the new entry into the table.
;  424     1574  2      ! This is accomplished by traversing the Image Section Table Entries, looking for any
;  425     1575  2      ! Non-Based Global or Stack ISDs which follow the last "Normal" ISD.  When this location
;  426     1576  2      ! is found, the links in the affected ISEs are modified to include the new PATCH ISE.
;  427     1577  2      !--
;  428     1578  2      LOCAL_ISE_PTR = .PAT$GL_ISELHD;                                 ! Get the list head.
;  429     1579  2      PREV_ISE_PTR = .LOCAL_ISE_PTR;                                  ! Set PREV = Current for first pass.
;  430     1580  2      ISD_PTR = CH$PTR (.LOCAL_ISE_PTR, ISE$C_SIZE);                  ! Point to the first ISD in the list.
;  431     1581  2
;  432     1582  3      UNTIL ( (.LOCAL_ISE_PTR EQL 0) OR
;  433     1583  3              (.ISD_PTR[ISD$B_TYPE] EQL ISD$K_USRSTACK) OR
;  434     1584  2              (.ISD_PTR[ISD$V_GBL] AND NOT .ISD_PTR[ISD$V_BASED]) ) DO
;  435     1585  3              BEGIN
;  436     1586  3              IF NOT NULLPARAMETER (5)                                ! Was an address included in the call?
;  437     1587  3              THEN                                                    ! If so, then check to see if it maps
;  438     1588  3                  IF  .ADR GEQ .ISD_PTR[ISD$V_VPN] ^9                 ! into this ISD.
;  439     1589  5                  AND .ADR LEQ ((.ISD_PTR[ISD$V_VPN] +
;  440     1590  3                                .ISD_PTR[ISD$W_PAGCNT]) ^9) - 1
;  441     1591  3                  THEN                                                ! It does map into this ISD, so save the
;  442     1592  4                      BEGIN                                           ! attributes for the new default patch area
;  443     1593  4                      PFC =    .ISD_PTR [ISD$B_PFC];                  ! Page Fault Cluster size.
;  444     1594  4                      FLAGS = .ISD_PTR [ISD$L_FLAGS];                 ! Image section Flags.
;  445     1595  4                      TYPE =  .ISD_PTR [ISD$B_TYPE];                  ! Image section Type.
;  446     1596  4                      IDENT = .ISD_PTR [ISD$L_IDENT];                 ! Image section Ident.
;  447     1597  3                      END;
;  448     1598  3              PREV_ISE_PTR = .LOCAL_ISE_PTR;                          ! Save the address of the just checked ISE.
;  449     1599  3              LOCAL_ISE_PTR = .LOCAL_ISE_PTR[ISE$L_NXTISE];           ! Advance the pointer to the next ISE.
;  450     1600  3              ISD_PTR = CH$PTR (.LOCAL_ISE_PTR, ISE$C_SIZE);          ! Point to the next ISD also.
;  451     1601  2              END;
;  452     1602  2
;  453     1603  2      !++
;  454     1604  2      ! At this point we should be positioned to the location for inserting the new PATCH ISE/ISD pair.
;  455     1605  2      !--
;  456     1606  2      LOCAL_ISE_PTR = CH$PTR (..ISE_PTR, 0);                          ! Pick up the address of the new ISE.
;  457     1607  2      TEMP = .PREV_ISE_PTR[ISE$L_NXTISE];                             ! Save the FLINK to next ISE.
;  458     1608  2      PREV_ISE_PTR[ISE$L_NXTISE] = .LOCAL_ISE_PTR;                    ! Set FLINK to the new ISE.
;  459     1609  2      LOCAL_ISE_PTR[ISE$L_NXTISE] = .TEMP;                            ! Remember to point to the next ISE.
;  460     1610  2
;  461     1611  2      !++
;  462     1612  2      ! Initialize the image section table information.
;  463     1613  2      !--
;  464     1614  2      LOCAL_ISE_PTR[ISE$L_MAPVST] = 0;
;  465     1615  2      LOCAL_ISE_PTR[ISE$L_MAPVEND] = 0;
```

```
;  466     1616  2 LOCAL_ISE_PTR[ISE$L_IMGVST] = 0;
;  467     1617  2 LOCAL_ISE_PTR[ISE$L_IMGVEND] = 0;
;  468     1618  2
;  469     1619  2 !++
;  470     1620  2 ! Now build the image section descriptor.
;  471     1621  2 !--
;  472     1622  2 ISD_PTR = CH$PTR(.LOCAL_ISE_PTR, ISE$C_SIZE);                    ! Point to ISD
;  473     1623  2 ! ****** THIS SHOULD CHANGE WHEN IDENT FIELD IS DEFINED FOR PROCESS PRIVATE IMAGE SECTIONS.
;  474     1624  2 ! ISD_PTR[ISD$W_SIZE] = (IF (PAT$K_LENPRIV GTR ISD$K_LENPRIV) THEN PAT$K_LENPRIV ELSE ISD$K_LENPRIV);
;  475     1625  2 ! *****
;  476     1626  2 ISD_PTR[ISD$W_SIZE] = ISD$K_LENPRIV;
;  477     1627  2 ISD_PTR[ISD$W_PAGCNT] = .PAGE_CNT;
;  478     1628  2 ISD_PTR[ISD$L_VPNPFC] = .VPN;
;  479     1629  2 ISD_PTR[ISD$B_PFC] = 0;
;  480     1630  2 ISD_PTR[ISD$L_FLAGS] = 0;
;  481     1631  2 ISD_PTR[ISD$V_CRF] = TRUE;
;  482     1632  2 ISD_PTR[ISD$V_WRT] = TRUE;
;  483     1633  2 ISD_PTR[ISD$V_MATCHCTL] = ISD$K_MATNEV;
;  484     1634  2 ISD_PTR[ISD$B_TYPE] = ISD$K_NORMAL;
;  485     1635  2 ISD_PTR[ISD$L_VBN] = .VBN;
;  486     1636  2 ISD_PTR[ISD$L_IDENT] = 0;
;  487     1637  2
;  488     1638  2 IF NOT NULLPARAMETER (5)                                       ! Should we propagate the "patched"
;  489     1639  2 THEN                                                          ! image section attributes?
;  490     1640  3     BEGIN
;  491     1641  3     ISD_PTR[ISD$B_PFC] = .PFC;
;  492     1642  3     ISD_PTR[ISD$L_FLAGS] = .FLAGS;
;  493     1643  3     ISD_PTR[ISD$B_TYPE] = .TYPE;
;  494     1644  3     ISD_PTR[ISD$L_IDENT] = .IDENT;
;  495     1645  2     END;
;  496     1646  2
;  497     1647  2 RETURN;
;  498     1648  1 END;                                                           ! End of PAT$BUILD_ISE


                                                  01FC 00000          .ENTRY  PAT$BUILD_ISE, Save R2,R3,R4,R5,R6,R7,R8    ; 1502
                                          04      AC   DD 00002       PUSHL   ISE_PTR                                     ; 1568
                                                  28   DD 00005       PUSHL   #40
                                00000000G         02   FB 00007       CALLS   #2, PAT$ALLOBLK
                                          51 00000000G  EF   D0 0000E  MOVL    PAT$GL_ISELHD, LOCAL_ISE_PTR            ; 1578
                                          53            51   D0 00015  MOVL    LOCAL_ISE_PTR, PREV_ISE_PTR             ; 1579
                                          50      14    A1   9E 00018 1$:    MOVAB   20(R1), ISD_PTR                     ; 1580
                                                  51   D5 0001C       TSTL    LOCAL_ISE_PTR                           ; 1582
                                                  5B   13 0001E       BEQL    4$
                                       FD 8F      0B    A0   91 00020       CMPB    11(ISD_PTR), #253                   ; 1583
                                                  54   13 00025       BEQL    4$
                                          05      08    A0   E9 00027       BLBC    8(ISD_PTR), 2$                      ; 1584
                                    4B    09      A0    01   E1 0002B       BBC     #1, 9(ISD_PTR), 4$
                                          05            6C   91 00030 2$:    CMPB    (AP), #5                            ; 1586
                                                  3E   1F 00033       BLSSU   3$
                                          14      AC    D5   00035       TSTL    20(AP)
                                                  39   13 00038       BEQL    3$
                                52        04    A0    15   00   EF 0003A       EXTZV   #0, #21, 4(ISD_PTR), R2             ; 1588
                                                52      52   09   78 00040       ASHL    #9, R2, R2                          ;
```

```
                        52      14  AC  D1 00044          CMPL    ADR, R2                                                 ;
                                    29  19 00048          BLSS    3$                                                      ;
        52      04  A0          15      EF 0004A          EXTZV   #0, #21, 4(ISD_PTR), R2                                 ; 1590
                        58  02  A0  3C 00050          MOVZWL  2(ISD_PTR), R8                                          ;
                        52      58  C0 00054          ADDL2   R8, R2                                                  ;
                    52          52      09  78 00057          ASHL    #9, R2, R2                                              ;
                                    52  D7 0005B          DECL    R2                                                      ;
                        52      14  AC  D1 0005D          CMPL    ADR, R2                                                 ;
                                    10  14 00061          BGTR    3$                                                      ;
                        57  07  A0  90 00063          MOVB    7(ISD_PTR), PFC                                         ; 1593
                        55  08  A0  D0 00067          MOVL    8(ISD_PTR), FLAGS                                       ; 1594
                        56  0B  A0  90 0006B          MOVB    11(ISD_PTR), TYPE                                       ; 1595
                        54  10  A0  D0 0006F          MOVL    16(ISD_PTR), IDENT                                      ; 1596
                        53      51  D0 00073  3$:     MOVL    LOCAL_ISE_PTR, PREV_ISE_PTR                             ; 1598
                        51      61  D0 00076          MOVL    (LOCAL_ISE_PTR), LOCAL_ISE_PTR                          ; 1599
                                    9D  11 00079          BRB     1$                                                      ; 1600
                        51  04  BC  D0 0007B  4$:     MOVL    @ISE_PTR, LOCAL_ISE_PTR                                 ; 1606
                        52      63  D0 0007F          MOVL    (PREV_ISE_PTR), TEMP                                    ; 1607
                        63      51  D0 00082          MOVL    LOCAL_ISE_PTR, (PREV_ISE_PTR)                           ; 1608
                        61      52  D0 00085          MOVL    TEMP, (LOCAL_ISE_PTR)                                   ; 1609
                            0C  A1  7C 00088          CLRQ    12(LOCAL_ISE_PTR)                                       ; 1614
                            04  A1  7C 0008B          CLRQ    4(LOCAL_ISE_PTR)                                        ; 1616
                        50  14  A1  9E 0008E          MOVAB   20(R1), ISD_PTR                                         ; 1622
                        60      10  B0 00092          MOVW    #16, (ISD_PTR)                                          ; 1626
                02  A0          10  AC  B0 00095          MOVW    PAGE_CNT, 2(ISD_PTR)                                    ; 1627
                04  A0          08  AC  D0 0009A          MOVL    VPN, 4(ISD_PTR)                                         ; 1628
                                07  A0  94 0009F          CLRB    7(ISD_PTR)                                              ; 1629
                        52      08  A0  9E 000A2          MOVAB   8(ISD_PTR), R2                                          ; 1630
                                    62  D4 000A6          CLRL    (R2)                                                    ;
                        62          0A  88 000A8          BISB2   #10, (R2)                                               ; 1632
        62      03          04      03  F0 000AB          INSV    #3, #4, #3, (R2)                                        ; 1633
                                0B  A0  94 000B0          CLRB    11(ISD_PTR)                                             ; 1634
                            0C  A0      0C  AC  D0 000B3          MOVL    VBN, 12(ISD_PTR)                                        ; 1635
                                10  A0  D4 000B8          CLRL    16(ISD_PTR)                                             ; 1636
                        05          6C  91 000BB          CMPB    (AP), #5                                                ; 1638
                                    14  1F 000BE          BLSSU   5$                                                      ;
                                14  AC  D5 000C0          TSTL    20(AP)                                                  ;
                                    0F  13 000C3          BEQL    5$                                                      ;
                            07  A0          57  90 000C5          MOVB    PFC, 7(ISD_PTR)                                         ; 1641
                        62          55  D0 000C9          MOVL    FLAGS, (R2)                                             ; 1642
                            0B  A0          56  90 000CC          MOVB    TYPE, 11(ISD_PTR)                                       ; 1643
                            10  A0          54  D0 000D0          MOVL    IDENT, 16(ISD_PTR)                                      ; 1644
                                            04  000D4  5$:     RET                                                             ; 1648

; Routine Size:  213 bytes,    Routine Base:  _PAT$CODE + 0111
```

```
;  500     1649  1 GLOBAL ROUTINE PAT$EXP_AREA (NUM_BLKS, ADR) : NOVALUE =          ! Expands patch area
;  501     1650  1
;  502     1651  1 !++
;  503     1652  1 ! FUNCTIONAL DESCRIPTION:
;  504     1653  1 !
;  505     1654  1 !       This routine expands the read-write patch area defined in the image
;  506     1655  1 !       header.  If there is no patch area, then an image section descriptor
;  507     1656  1 !       is created for it.  If the image section which is being created is a
;  508     1657  1 !       due to either an INSERT or REPLACE command then the attributes of the
;  509     1658  1 !       image section are propagated to the new image section.  In either case,
;  510     1659  1 !       the image header is updated to describe the expanded patch area.
;  511     1660  1 !
;  512     1661  1 !       If the patch area is mapped to the highest address used during this
;  513     1662  1 !       patch session, then the patch area can be expanded contiguously.
;  514     1663  1 !       In this case, the image section descriptor is updated to hold a new
;  515     1664  1 !       page count and the patch area size in the image header is increased.
;  516     1665  1 !       If the patch area is not the highest address used, then the patch area
;  517     1666  1 !       must be relocated to another area, which will be contiguous.  This
;  518     1667  1 !       involves expanding the program region, copying in the old patch area,
;  519     1668  1 !       and then changing the image section table entry to point to a new
;  520     1669  1 !       mapped address.  The image header and image section descriptor counts
;  521     1670  1 !       are incremented as above.
;  522     1671  1 !
;  523     1672  1 !       NOTE:  The patch area must be mapped contiguously in order for
;  524     1673  1 !       the mapping of addresses to work.  It could also be accomplished
;  525     1674  1 !       if two image section table entries were created.  However, this
;  526     1675  1 !       would require an extra, unneccessary image section descriptor.
;  527     1676  1 !
;  528     1677  1 !       Some of the PATCH commands which deposit symbolic instructions do an
;  529     1678  1 !       PAT$EXPAREA just to force the address to be non-zero so that
;  530     1679  1 !       the symbolic instruction encoder can correctly encode operands.
;  531     1680  1 !
;  532     1681  1 ! FORMAL PARAMETERS:
;  533     1682  1 !
;  534     1683  1 !       NUM_BLKS - Number of blocks to be allocated for the patch area
;  535     1684  1 !       ADR -[OPT] The address which we will use to propagate the image section
;  536     1685  1 !                  attributes.
;  537     1686  1 !
;  538     1687  1 ! IMPLICIT INPUTS:
;  539     1688  1 !
;  540     1689  1 !       The image header and image section entry table must be set up.
;  541     1690  1 !
;  542     1691  1 ! IMPLICIT OUTPUTS:
;  543     1692  1 !
;  544     1693  1 !       none
;  545     1694  1 !
;  546     1695  1 ! ROUTINE VALUE:
;  547     1696  1 !
;  548     1697  1 !       none
;  549     1698  1 !
;  550     1699  1 ! COMPLETION CODES:
;  551     1700  1 !
;  552     1701  1 !       none
;  553     1702  1 !
;  554     1703  1 ! SIDE EFFECTS:
;  555     1704  1 !
;  556     1705  1 !       A new patch area is set up.  The image header is updated to
```

```
;  557     1706  1 !       describe the new patch area.
;  558     1707  1 !
;  559     1708  1 !       ** If the file is being patched in absolute mode, we cannot
;  560     1709  1 !       expand the file (it would more than likely corrupt it!).                }
;  561     1710  1 !       In this instance, we'll abort the command back to the patch
;  562     1711  1 !       command prompt via an error severity signal.
;  563     1712  1 !
;  564     1713  1 !--
;  565     1714  1
;  566     1715  2 BEGIN
;  567     1716  2
;  568     1717  2 BUILTIN
;  569     1718  2         NULLPARAMETER;
;  570     1719  2
;  571     1720  2 LITERAL
;  572     1721  2         START_OFF = 0,                                          ! Offset to starting address
;  573     1722  2         END_OFF = 1;                                            ! Offset to ending address
;  574     1723  2
;  575     1724  2 LOCAL
;  576     1725  2         ISE_PTR : REF BLOCK[,BYTE],                             ! Pointer to image section table entry
;  577     1726  2         ISD_PTR : REF BLOCK[,BYTE],                             ! Pointer to image section descriptor
;  578     1727  2         MAPPED_ADDR;                                            ! Mapped address
;  579     1728  2
;  580     1729  2 IF .PAT$GL_FLAGS [PAT$S_ABSOLUTE]                               ! If we're in absolute mode, then someone ha
;  581     1730  2 THEN    SIGNAL (PAT$_DATTOOLNG);                                ! us too long a datum, and the file was expe
;  582     1731  2                                                                 ! to be expanded (a no no!), return to promp
;  583     1732  2
;  584     1733  2 !++
;  585     1734  2 ! If this is a non-PIC shareable image we do not expand the patch area to protect images
;  586     1735  2 ! previously linked against having inconsistent Global Section Descriptors.  Else, if it
;  587     1736  2 ! is a PIC shareable image, we may without reservation, expand the patch area.
;  588     1737  2 !--
;  589     1738  3 IF ((.PAT$GL_IMGHDR[IHD$B_IMGTYPE] EQLU IHD$K_LIM) AND (NOT .PAT$GL_IMGHDR[IHD$V_PICIMG]))
;  590     1739  2 THEN
;  591     1740  2         SIGNAL(PAT$_EXPSHRPAT+MSG$K_SEVERE);
;  592     1741  2
;  593     1742  2 !++
;  594     1743  2 ! If there is no patch area defined yet, then build an image section table
;  595     1744  2 ! entry and an image section descriptor for it.
;  596     1745  2 !--
;  597     1746  3 IF (.PAT$GL_IHPPTR[IHP$L_RW_PATADR] EQLA 0)
;  598     1747  2 THEN
;  599     1748  3         BEGIN
;  600     1749  3         !++
;  601     1750  3         ! Build an Image Section table entry as no Patch Area was defined.
;  602     1751  3         !--
;  603     1752  3         IF NULLPARAMETER (2)
;  604     1753  3         THEN PAT$BUILD_ISE(ISE_PTR, .PAT$GL_NEWVPNMX+1, .PAT$GL_NEWVBNMX+1, .NUM_BLKS)
;  605     1754  3         ELSE PAT$BUILD_ISE(ISE_PTR, .PAT$GL_NEWVPNMX+1, .PAT$GL_NEWVBNMX+1, .NUM_BLKS, .ADR);
;  606     1755  3         ISD_PTR = CH$PTR(.ISE_PTR, ISE$C_SIZE);
;  607     1756  3         END
;  608     1757  2 ELSE
;  609     1758  3         BEGIN
;  610     1759  3         !++
;  611     1760  3         ! Find the image section table entry which describes the patch area.
;  612     1761  3         !--
;  613     1762  3         PAT$MAP_ADDR(.PAT$GL_IHPPTR[IHP$L_RW_PATADR], MAPPED_ADDR, ISE_PTR);
```

```
;  614          1763  3          ISD_PTR = CH$PTR(.ISE_PTR, ISE$C_SIZE);
;  615          1764  3          ISD_PTR[ISD$W_PAGCNT] = .ISD_PTR[ISD$W_PAGCNT] + .NUM_BLKS; ! Expand size of image section
;  616          1765  2          END;
;  617          1766  2
;  618          1767  2      !++
;  619          1768  2      ! Update the VPN and VBN for the last ones used in the new image for
;  620          1769  2      ! the image section.
;  621          1770  2      !--
;  622          1771  2      PAT$GL_NEWVPNMX = .PAT$GL_NEWVPNMX + .NUM_BLKS;
;  623          1772  2      PAT$GL_NEWVBNMX = .PAT$GL_NEWVBNMX + .NUM_BLKS;
;  624          1773  2
;  625          1774  2      !++
;  626          1775  2      ! Now create the patch area, i.e., map it into the image.  This is done
;  627          1776  2      ! with an expand region instead of a create and map as the area is not defined
;  628          1777  2      ! in the old image.
;  629          1778  2      !--
;  630        P 1779  2      PAT$GL_ERRCODE = $EXPREG(PAGCNT = .ISD_PTR[ISD$W_PAGCNT]
;  631          1780  2                               , RETADR = PAT$GL_ISVADDR);
;  632          1781  2      IF NOT .PAT$GL_ERRCODE
;  633          1782  2      THEN
;  634          1783  2          SIGNAL(PAT$_SYSERROR, 0, .PAT$GL_ERRCODE);
;  635          1784  2
;  636          1785  2      !++
;  637          1786  2      ! If the patch area was expanded, and not created, then copy in the old
;  638          1787  2      ! patch area part.
;  639          1788  2      !--
;  640          1789  3      IF (.ISD_PTR[ISD$W_PAGCNT] NEQ .NUM_BLKS)
;  641          1790  2      THEN
;  642          1791  2          CH$MOVE((.ISD_PTR[ISD$W_PAGCNT] - .NUM_BLKS) * A_PAGE,
;  643          1792  2                  .ISE_PTR[ISE$L_MAPVST], .PAT$GL_ISVADDR[START_OFF]);
;  644          1793  2
;  645          1794  2      !++
;  646          1795  2      ! Initialize the image section table entry.
;  647          1796  2      !--
;  648          1797  2      ISE_PTR[ISE$L_MAPVST] = .PAT$GL_ISVADDR[START_OFF];
;  649          1798  2      ISE_PTR[ISE$L_MAPVEND] = .PAT$GL_ISVADDR[END_OFF];
;  650          1799  2      ISE_PTR[ISE$L_IMGVST] = .ISD_PTR[ISD$V_VPN] ^9;
;  651          1800  2      ISE_PTR[ISE$L_IMGVEND] = ((.ISD_PTR[ISD$V_VPN] + .ISD_PTR[ISD$W_PAGCNT]) ^9) - 1;
;  652          1801  2
;  653          1802  2      !++
;  654          1803  2      ! Increment the number of blocks in the new image.
;  655          1804  2      !--
;  656          1805  2      PAT$GL_IMGBLKS = .PAT$GL_IMGBLKS + .NUM_BLKS;
;  657          1806  2
;  658          1807  2      !++
;  659          1808  2      ! Update the patch area descriptor in the image header.
;  660          1809  2      !--
;  661          1810  2      PAT$GL_PATAREA[DSC$W_LENGTH] = .PAT$GL_PATAREA[DSC$W_LENGTH] + (.NUM_BLKS * A_PAGE);
;  662          1811  3      IF (.PAT$GL_PATAREA[DSC$A_POINTER] EQLA 0)
;  663          1812  2      THEN
;  664          1813  2          PAT$GL_PATAREA[DSC$A_POINTER] = .ISE_PTR[ISE$L_IMGVST];
;  665          1814  2
;  666          1815  2      !++
;  667          1816  2      ! Now update the patch area list entry for the default patch area.
;  668          1817  2      !--
;  669          1818  2      PAT$ADD_PAL(.ISE_PTR[ISE$L_IMGVST], .ISE_PTR[ISE$L_IMGVEND], PAL$K_EXP_PAREA);
;  670          1819  2
```

```
;  671          1820  2 RETURN;
;  672          1821  2
;  673          1822  1 END;                                                    ! END OF PAT$EXP_AREA


                                                        .EXTRN  SYS$EXPREG

                                        0FFC 00000      .ENTRY  PAT$EXP_AREA, Save R2,R3,R4,R5,R6,R7,R8,R9,-; 1649
                                                                R10,R11
                                5B 00000000G   EF  9E 00002      MOVAB   PAT$GL_NEWVBNMX, R11
                                5A 00000000G   EF  9E 00009      MOVAB   PAT$GL_ERRCODE, R10
                                59 00000000G   EF  9E 00010      MOVAB   PAT$GL_ISVADDR, R9
                                58 00000000G   00  9E 00017      MOVAB   LIB$SIGNAL, R8
                                5E             08  C2 0001E      SUBL2   #8, SP
                     09 00000000G   EF         06  E1 00021      BBC     #6, PAT$GL_FLAGS, 1$                    1729
                                    006D80A2   8F  DD 00029      PUSHL   #7176354                                1730
                                68             01  FB 0002F      CALLS   #1, LIB$SIGNAL
                                50 00000000G   EF  D0 00032 1$:  MOVL    PAT$GL_IMGHDR, R0                       1738
                                02       11    A0  91 00039      CMPB    17(R0), #2
                                               0E  12 0003D      BNEQ    2$
                     09       20    A0         03  E0 0003F      BBS     #3, 32(R0), 2$
                                    006D82D2   8F  DD 00044      PUSHL   #7176914                                1740
                                68             01  FB 0004A      CALLS   #1, LIB$SIGNAL
                                57       04    AC  D0 0004D 2$:  MOVL    NUM_BLKS, R7                          1753
                                50 00000000G   EF  D0 00051      MOVL    PAT$GL_IHPPTR, R0                       1746
                                         14    A0  D5 00058      TSTL    20(R0)
                                               39  12 0005B      BNEQ    6$
                     51         68             01  C1 0005D      ADDL3   #1, PAT$GL_NEWVBNMX, R1                 1753
                     50 00000000G   EF         01  C1 00061      ADDL3   #1, PAT$GL_NEWVPNMX, R0
                                02             6C  91 00069      CMPB    (AP), #2                                1752
                                               05  1F 0006C      BLSSU   3$
                                         08    AC  D5 0006E      TSTL    8(AP)
                                               0E  12 00071      BNEQ    4$
                                       0083    8F  BB 00073 3$:  PUSHR   #^M<R0,R1,R7>                           1753
                                         0C    AE  9F 00077      PUSHAB  ISE_PTR
                                FEAC   CF          04  FB 0007A      CALLS   #4, PAT$BUILD_ISE
                                               0F  11 0007F      BRB     5$
                                         08    AC  DD 00081 4$:  PUSHL   ADR                                     1754
                                       0083    8F  BB 00084      PUSHR   #^M<R0,R1,R7>
                                         10    AE  9F 00088      PUSHAB  ISE_PTR
                                FE9B   CF          05  FB 0008B      CALLS   #5, PAT$BUILD_ISE
                     56         6E             14  C1 00090 5$:  ADDL3   #20, ISE_PTR, ISD_PTR                  1755
                                               17  11 00094      BRB     7$                                      1746
                                               5E  DD 00096 6$:  PUSHL   SP                                      1762
                                         08    AE  9F 00098      PUSHAB  MAPPED_ADDR
                                         14    A0  DD 0009B      PUSHL   20(R0)
                          00000000G    EF         03  FB 0009E      CALLS   #3, PAT$MAP_ADDR
                     56         6E             14  C1 000A5      ADDL3   #20, ISE_PTR, ISD_PTR                  1763
                                   02    A6  57  A0 000A9      ADDW2   R7, 2(ISD_PTR)                          1764
                          00000000G    EF         57  C0 000AD 7$:  ADDL2   R7, PAT$GL_NEWVPNMX                  1771
                                68             57  C0 000B4      ADDL2   R7, PAT$GL_NEWVBNMX                     1772
                                7E             7C  000B7      CLRQ    -(SP)                                   1780
                                59             DD  000B9      PUSHL   R9
                                7E       02    A6  3C 000BB      MOVZWL  2(ISD_PTR), -(SP)
                          00000000G    00         04  FB 000BF      CALLS   #4, SYS$EXPREG
                                6A             50  D0 000C6      MOVL    R0, PAT$GL_ERRCODE
```

```
                                 0D              6A  E8 000C9          BLBS    PAT$GL_ERRCODE, 8$                          ; 1781
                                                 6A  DD 000CC          PUSHL   PAT$GL_ERRCODE                              ; 1783
                                                 7E  D4 000CE          CLRL    -(SP)
                           00000000G             8F  DD 000D0          PUSHL   #PAT$_SYSERROR
                                 68              03  FB 000D6          CALLS   #3, LIB$SIGNAL
    57      02    A6             10              00  ED 000D9 8$:      CMPZV   #0, #16, 2(ISD_PTR), R7                     ; 1789
                                                 14  13 000DF          BEQL    9$
                                 51      02      A6  3C 000E1          MOVZWL  2(ISD_PTR), R1                              ; 1791
                                 51              57  C2 000E5          SUBL2   R7, R1
                       51        51              09  78 000E8          ASHL    #9, R1, R1
                                 50              6E  D0 000EC          MOVL    ISD_PTR, R0                                 ; 1792
            00    B9       0C    B0              51  28 000EF          MOVC3   R1, @12(R0), @PAT$GL_ISVADDR
                                 50              6E  D0 000F5 9$:      MOVL    ISD_PTR, R0                                 ; 1797
                           0C    A0              69  7D 000F8          MOVQ    PAT$GL_ISVADDR, 12(R0)
    51      04    A6             15              00  EF 000FC          EXTZV   #0, #21, 4(ISD_PTR), R1                     ; 1799
            04    A0             51              09  78 00102          ASHL    #9, R1, 4(R0)
    51      04    A6             15              00  EF 00107          EXTZV   #0, #21, 4(ISD_PTR), R1                     ; 1800
                                 56      02      A6  3C 0010D          MOVZWL  2(ISD_PTR), R6
                                 56              51  C0 00111          ADDL2   R1, R6
                       56        56              09  78 00114          ASHL    #9, R6, R6
                           08    A0      FF      A6  9E 00118          MOVAB   -1(R6), 8(R0)
                00000000G        EF              57  C0 0011D          ADDL2   R7, PAT$GL_IMGBLKS                          ; 1805
                                 51  00000000G   EF  D0 00124          MOVL    PAT$GL_PATAREA, R1                          ; 1810
                       52        57              09  78 0012B          ASHL    #9, R7, R2
                                 61              52  A0 0012F          ADDW2   R2, (R1)
                                         04      A1  D5 00132          TSTL    4(R1)                                       ; 1811
                                                 05  12 00135          BNEQ    10$
                           04    A1      04      A0  D0 00137          MOVL    4(R0), 4(R1)                                ; 1813
                                                 01  DD 0013C 10$:     PUSHL   #1                                          ; 1818
                                 7E      04      A0  7D 0013E          MOVQ    4(R0), -(SP)
                00000000V        EF              03  FB 00142          CALLS   #3, PAT$ADD_PAL
                                                 04  00149          RET                                                 ; 1822

; Routine Size:  330 bytes,    Routine Base:  _PAT$CODE + 01E6
```

```
;  675     1823  1 GLOBAL ROUTINE PAT$ADD_PAL (START_ADR, END_ADR, PAT_AREA_FLAG) : NOVALUE =      ! EXPANDS PATCH AREAS
;  676     1824  1
;  677     1825  1 !++
;  678     1826  1 ! FUNCTIONAL DESCRIPTION:
;  679     1827  1 !
;  680     1828  1 !       THIS ROUTINE MAINTAINS THE PATCH AREA LIST (PAL).  THIS INCLUDES
;  681     1829  1 !       UPDATING THE ENTRY FOR THE DEFAULT PATCH AREA WHENEVER PATCH EXPANDS
;  682     1830  1 !       IT AND CREATING ENTRIES WHENEVER THE USER ISSUES A "SET PATCH_AREA"
;  683     1831  1 !       COMMAND.  THE FIRST ENTRY ON THE LIST IS ALWAYS THE DEFAULT PATCH AREA.
;  684     1832  1 !
;  685     1833  1 !       THE PATCH AREA LIST IS USED TO CORRECTLY OUTPUT ADDRESSES FOR
;  686     1834  1 !       PATCH AREA TO THE OUTPUT COMMAND FILE.  THESE ADDRESSES MUST BE
;  687     1835  1 !       WRITTEN TO THE FILE AS SYMBOLIC NAMES PLUS OFFSETS BECAUSE THE
;  688     1836  1 !       IMAGES IN THE FIELD MAY HAVE BEEN PATCHED BY CUSTOMERS ( THUS
;  689     1837  1 !       CHANGING THE NEXT FREE PATCH AREA ADDRESS ).  BY OUTPUTING PATCH
;  690     1838  1 !       AREA ADDRESSES AS SYMBOLIC NAMES, PATCH WILL PERMIT PATCHES TO
;  691     1839  1 !       USE DIFFERENT PATCH AREA ADDRESSES.
;  692     1840  1 !
;  693     1841  1 !       AN ENTRY IN THE PATCH AREA LIST HAS THE FOLLOWING FORMAT:
;  694     1842  1 !
;  695     1843  1 !               +-----------------------------+
;  696     1844  1 !               !        FORWARD LINK         !  PAL$L_FLINK
;  697     1845  1 !               +-----------------------------+
;  698     1846  1 !               !      STARTING ADDRESS       !  PAL$L_ST_ADR
;  699     1847  1 !               +-----------------------------+
;  700     1848  1 !               !       ENDING ADDRESS        !  PAL$L_END_ADR
;  701     1849  1 !               +-----------------------------+
;  702     1850  1 !               !      PATCH AREA NAME        !  PAL$L_CS_NAME
;  703     1851  1 !               +-----------------------------+
;  704     1852  1 !
;  705     1853  1 !       THE PATCH AREA NAME CONSISTS OF AN ASCIC STRING, WHICH IS ALWAYS A
;  706     1854  1 !       COUNT OF THREE FOLLOWED BY THE ASCII CHARACTERS "P", "A", AND A THIRD
;  707     1855  1 !       CHARACTER RANGING FROM "A" TO "Z".  THIS NAME IS USED TO OUTPUT
;  708     1856  1 !       SYMBOLIC REFERENCES TO THE OUTPUT COMMAND FILE FOR ALL ADDRESSES WITHIN
;  709     1857  1 !       THE PATCH AREAS INSTEAD OF ABSOLUTE VALUES.
;  710     1858  1 !
;  711     1859  1 !       THIS ROUTINE ALSO CAUSES A SYMBOL TO BE DEFINED FOR THE STARTING ADDRESS
;  712     1860  1 !       OF THE PATCH AREA.
;  713     1861  1 !
;  714     1862  1 ! FORMAL PARAMETERS:
;  715     1863  1 !
;  716     1864  1 !       START_ADR - STARTING ADDRESS OF THE PATCH AREA
;  717     1865  1 !       END_ADR - ENDING ADDRESS OF THE PATCH AREA
;  718     1866  1 !       PAT_AREA_FLAG - INDICATOR FOR TYPE OF PAL UPDATE
;  719     1867  1 !                       PAL$K_EXP_PAREA = 1  - EXPANDING DEFAULT PATCH AREA
;  720     1868  1 !                       PAL$K_ADD_PAREA = 0  - ADDING NEW PATCH AREA ENTRY
;  721     1869  1 !
;  722     1870  1 ! IMPLICIT INPUTS:
;  723     1871  1 !
;  724     1872  1 !       THE FREE STORAGE ROUTINES MUST HAVE BEEN INITIALIZED.
;  725     1873  1 !
;  726     1874  1 ! IMPLICIT OUTPUTS:
;  727     1875  1 !
;  728     1876  1 !       NONE
;  729     1877  1 !
;  730     1878  1 ! ROUTINE VALUE:
;  731     1879  1 !
```

```
;  732     1880  1 !       NONE
;  733     1881  1 !
;  734     1882  1 ! COMPLETION CODES:
;  735     1883  1 !
;  736     1884  1 !       NONE
;  737     1885  1 !
;  738     1886  1 ! SIDE EFFECTS:
;  739     1887  1 !
;  740     1888  1 !       THE PATCH AREA LIST IS UPDATED.  EITHER AN ENTRY IS MODIFIED OR
;  741     1889  1 !       A NEW LINK IS CREATED.  IN THE LATTER CASE, THE NEXT PATCH AREA NAME
;  742     1890  1 !       IS ALSO UPDATED.  THE NEXT PATCH AREA NAME IS ALSO UPDATED.
;  743     1891  1 !
;  744     1892  1 !--
;  745     1893  1
;  746     1894  2 BEGIN
;  747     1895  2
;  748     1896  2 LOCAL
;  749     1897  2         TEMP_SYMTB,                                     ! Temporary symbol table pointer
;  750     1898  2         NEW_PTR : REF BLOCK[,BYTE],                     ! POINTER TO NEW PAL ENTRY
;  751     1899  2         TEMP_PTR : REF BLOCK[,BYTE],                    ! POINTER TO CURRENT PAL ENTRY
;  752     1900  2         NAME_DESC : BLOCK[8,BYTE];                      ! STRING DESCRIPTOR FOR DEFAULT PATCH AREA N
;  753     1901  2
;  754     1902  2 !++
;  755     1903  2 ! FIRST, LOOP THROUGH THE PATCH AREA LIST TRYING TO FIND AN ENTRY FOR THIS
;  756     1904  2 ! PATCH AREA, I.E., HAS THIS PATCH AREA JUST BEEN EXPANDED.  IF SO, UPDATE
;  757     1905  2 ! THE PAL ENTRY AND RETURN.  IF NOT, FALL THROUGH TO CREATE A NEW PAL ENTRY.
;  758     1906  2 !--
;  759     1907  2 TEMP_SYMTB = .PAT$GL_SYMTBPTR;                          ! Remember current label symbol table
;  760     1908  2 IF (TEMP_PTR = CH$PTR(.PAT$GL_PAL_LHD, 0)) NEQ 0        ! GET FIRST ENTRY IN LIST
;  761     1909  2 THEN
;  762     1910  2         REPEAT
;  763     1911  3                 BEGIN
;  764     1912  3                 !++
;  765     1913  3                 ! IF THE DEFAULT PATCH AREA WAS CREATED, THEN BOTH THE STARTING
;  766     1914  3                 ! AND ENDING ADDRESSES MUST BE RESET.  IF THE DEFAULT PATCH
;  767     1915  3                 ! AREA WAS EXPANDED, THEN THE STARTING ADDRESS REMAINS THE
;  768     1916  3                 ! SAME AND THE ENDING ADDRESS IS UPDATED.  THIS WILL NEED
;  769     1917  3                 ! SOME NEW INVENTION WHEN READ-ONLY PATCH AREAS ARE
;  770     1918  3                 ! ALSO ADDED.
;  771     1919  3                 !--
;  772     1920  3                 IF .PAT_AREA_FLAG EQL PAL$K_EXP_PAREA
;  773     1921  3                 THEN
;  774     1922  4                         BEGIN
;  775     1923  4                         TEMP_PTR[PAL$L_END_ADR] = .END_ADR;
;  776     1924  4                         IF .TEMP_PTR[PAL$L_START_ADR] EQLA 0
;  777     1925  4                         THEN
;  778     1926  5                                 BEGIN
;  779     1927  5                                 TEMP_PTR[PAL$L_START_ADR] = .START_ADR;
;  780     1928  5                                 NAME_DESC[DSC$Q_LENGTH] = .PAT_AREA_NAME[0];
;  781     1929  5                                 NAME_DESC[DSC$A_POINTER] = CH$PTR(TEMP_PTR[PAL$L_CS_NAME], 1);
;  782     1930  5                                 PAT$GL_SYMTBPTR = .PAT$GL_SYMHEAD;
;  783     1931  5                                 PAT$DEFINE_SYM(NAME_DESC, .START_ADR, FALSE);
;  784     1932  5                                 PAT$GL_SYMTBPTR = .TEMP_SYMTB;
;  785     1933  4                                 END;
;  786     1934  4                         RETURN;
;  787     1935  3                         END;
;  788     1936  3                 IF (.START_ADR GEQA .TEMP_PTR[PAL$L_START_ADR]) AND
```

```
;  789          1937  4                  (.END_ADR EQLA .TEMP_PTR[PAL$L_END_ADR])
;  790          1938  3                THEN
;  791          1939  3                    RETURN;
;  792          1940  3                IF .TEMP_PTR[PAL$L_FLINK] NEQA 0
;  793          1941  3                THEN
;  794          1942  3                    TEMP_PTR = .TEMP_PTR[PAL$L_FLINK]
;  795          1943  3                ELSE
;  796          1944  3                    EXITLOOP;
;  797          1945  2                END;
;  798          1946  2
;  799          1947  2            !++
;  800          1948  2            ! THERE WAS NO CORRESPONDING PAL ENTRY.  THEREFORE A NEW ENTRY MUST BE CREATED.
;  801          1949  2            !--
;  802          1950  2            NEW_PTR = PAT$FREEZ((PAL$C_SIZE + A_LONGWORD - 1)/A_LONGWORD);  ! ALLOCATE SPACE FOR NEW ENTRY
;  803          1951  2            IF .TEMP_PTR EQLA 0
;  804          1952  2            THEN
;  805          1953  2                PAT$GL_PAL_LHD = CH$PTR(.NEW_PTR, 0)                        ! SET THE LIST HEAD
;  806          1954  2            ELSE
;  807          1955  2                TEMP_PTR[PAL$L_FLINK] = .NEW_PTR;                           ! LINK IN NEW ENTRY
;  808          1956  2            NEW_PTR[PAL$L_START_ADR] = .START_ADR;                          ! SET STARTING PATCH AREA ADDRESS
;  809          1957  2            NEW_PTR[PAL$L_END_ADR] = .END_ADR;                              ! SET ENDING PATCH AREA ADDRESS
;  810          1958  2            CH$MOVE(A_LONGWORD, PAT_AREA_NAME, NEW_PTR[PAL$L_CS_NAME]);     ! SET PATCH AREA NAME
;  811          1959  2            PAT$GL_SYMTBPTR = .PAT$GL_SYMHEAD;                              ! Use user-defined symbol table
;  812          1960  2            PAT$DEFINE_SYM(PA_NAME_DSC, .NEW_PTR[PAL$L_START_ADR], FALSE);  ! DEFINE SYMBOL AS START OF PATCH AREA
;  813          1961  2            PAT$GL_SYMTBPTR = .TEMP_SYMTB;                                  ! Restore label symbol table
;  814          1962  2            PAT_AREA_NAME[3] = .PAT_AREA_NAME[3] + 1;                       ! SET NEW PATCH AREA NAME
;  815          1963  2
;  816          1964  2            !++
;  817          1965  2            ! NOW CHECK THAT THE NEXT PATCH AREA NAME IS BETWEEN "PAA" AND "PAZ".  IF
;  818          1966  2            ! IT IS NOT, THE RESET THE THIRD CHARACTER OF THE NAME TO AN "A" AND
;  819          1967  2            ! INCREMENT THE SECOND LETTER OF THE NAME.  THIS WILL ALLOW THE USER TO DEFINE
;  820          1968  2            ! UP TO 676 PATCH AREAS.
;  821          1969  2            !--
;  822          1970  3            IF .PAT_AREA_NAME[3] GTRU (%ASCII'Z')                           ! CHECK FOR OVERFLOW OF PATCH AREA NAMES
;  823          1971  2            THEN
;  824          1972  3                BEGIN
;  825          1973  3                PAT_AREA_NAME[2] = .PAT_AREA_NAME[2] + 1;                   ! INCREMENT THE "A" OF "PAZ"
;  826          1974  3                PAT_AREA_NAME[3] = (%ASCII'A');                             ! CHANGE THE "Z" TO AN "A"
;  827          1975  2                END;
;  828          1976  2
;  829          1977  2            RETURN;
;  830          1978  2
;  831          1979  1            END;                                                            ! END OF PAT$ADD_PAL


                                    01FC 00000          .ENTRY  PAT$ADD_PAL, Save R2,R3,R4,R5,R6,R7,R8      ; 1823
                58 00000000G    EF  9E 00002        MOVAB   PAT$DEFINE_SYM, R8                          ;
                57 00000000G    EF  9E 00009        MOVAB   PAT$GL_SYMHEAD, R7                          ;
                56 00000000G    EF  9E 00010        MOVAB   PAT$GL_PAL_LHD, R6                          ;
                55 00000000G    EF  9E 00017        MOVAB   PAT$GL_SYMTBPTR, R5                         ;
                54 00000000'    EF  9E 0001E        MOVAB   PAT_AREA_NAME+3, R4                         ;
                5E              08  C2 00025        SUBL2   #8, SP                                      ;
                53              65  D0 00028        MOVL    PAT$GL_SYMTBPTR, TEMP_SYMTB                 ; 1907
                52              66  D0 0002B        MOVL    PAT$GL_PAL_LHD, TEMP_PTR                    ; 1908
```

```
                                47  13 0002E             BEQL    4$                                                      ; 1920
                        01      0C  AC  D1 00030 1$:      CMPL    PAT_AREA_FLAG, #1                                       ;
                                2A  12 00034             BNEQ    2$                                                      ;
                    08  A2      08  AC  D0 00036         MOVL    END_ADR, 8(TEMP_PTR)                                    ; 1923
                                04  A2  D5 0003B         TSTL    4(TEMP_PTR)                                             ; 1924
                                76  12 0003E             BNEQ    7$                                                      ;
                    04  A2      04  AC  D0 00040         MOVL    START_ADR, 4(TEMP_PTR)                                  ; 1927
                    6E  FD      A4  9B 00045             MOVZBW  PAT_AREA_NAME, NAME_DESC                                ; 1928
                    04  AE      0D  A2  9E 00049         MOVAB   13(R2), NAME_DESC+4                                     ; 1929
                        65          67  D0 0004E         MOVL    PAT$GL_SYMHEAD, PAT$GL_SYMTBPTR                         ; 1930
                                    7E  D4 00051         CLRL    -(SP)                                                   ; 1931
                                04  AC  DD 00053         PUSHL   START_ADR                                               ;
                                08  AE  9F 00056         PUSHAB  NAME_DESC                                               ;
                        68          03  FB 00059         CALLS   #3, PAT$DEFINE_SYM                                      ;
                        65          53  D0 0005C         MOVL    TEMP_SYMTB, PAT$GL_SYMTBPTR                             ; 1932
                                    04  0005F             RET                                                             ; 1922
                    04  A2      04  AC  D1 00060 2$:      CMPL    START_ADR, 4(TEMP_PTR)                                  ; 1936
                                07  1F 00065             BLSSU   3$                                                      ;
                    08  A2      08  AC  D1 00067         CMPL    END_ADR, 8(TEMP_PTR)                                    ; 1937
                                48  13 0006C             BEQL    7$                                                      ;
                                62  D5 0006E 3$:          TSTL    (TEMP_PTR)                                              ; 1940
                                05  13 00070             BEQL    4$                                                      ;
                        52          62  D0 00072         MOVL    (TEMP_PTR), TEMP_PTR                                    ; 1942
                                B9  11 00075             BRB     1$                                                      ;
                                04  DD 00077 4$:          PUSHL   #4                                                      ; 1950
    00000000G   EF              01  FB 00079             CALLS   #1, PAT$FREEZ                                           ;
                                52  D5 00080             TSTL    TEMP_PTR                                                ; 1951
                                05  12 00082             BNEQ    5$                                                      ;
                        66          50  D0 00084         MOVL    NEW_PTR, PAT$GL_PAL_LHD                                 ; 1953
                                03  11 00087             BRB     6$                                                      ;
                        62          50  D0 00089 5$:      MOVL    NEW_PTR, (TEMP_PTR)                                     ; 1955
                    04  A0      04  AC  7D 0008C 6$:      MOVQ    START_ADR, 4(NEW_PTR)                                   ; 1956
                    0C  A0      FD  A4  D0 00091         MOVL    PAT_AREA_NAME, 12(NEW_PTR)                              ; 1958
                        65          67  D0 00096         MOVL    PAT$GL_SYMHEAD, PAT$GL_SYMTBPTR                         ; 1959
                                    7E  D4 00099         CLRL    -(SP)                                                   ; 1960
                                04  A0  DD 0009B         PUSHL   4(NEW_PTR)                                              ;
                                01  A4  9F 0009E         PUSHAB  PA_NAME_DSC                                             ;
                        68          03  FB 000A1         CALLS   #3, PAT$DEFINE_SYM                                      ;
                        65          53  D0 000A4         MOVL    TEMP_SYMTB, PAT$GL_SYMTBPTR                             ; 1961
                                    64  96 000A7         INCB    PAT_AREA_NAME+3                                         ; 1962
                    5A  8F              64  91 000A9     CMPB    PAT_AREA_NAME+3, #90                                    ; 1970
                                    07  1B 000AD         BLEQU   7$                                                      ;
                                FF  A4  96 000AF         INCB    PAT_AREA_NAME+2                                         ; 1973
                        64      41  8F  90 000B2         MOVB    #65, PAT_AREA_NAME+3                                    ; 1974
                                    04  000B6 7$:        RET                                                             ; 1979

; Routine Size:  183 bytes,    Routine Base:  _PAT$CODE + 0330
```

```
;  833          1980  1 END                                   ! End of module
;  834          1981  0 ELUDOM


                                                    .EXTRN  LIB$SIGNAL

;                               PSECT SUMMARY

;       Name                    Bytes                   Attributes

;   _PAT$OWN                      12  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   _PAT$PLIT                    116  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(0)
;   _PAT$CODE                    999  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   .  ABS  .                      0  NOVEC,NOWRT,NORD ,NOEXE,NOSHR,  LCL,  ABS,  CON,NOPIC,ALIGN(0)


;                           Library Statistics

;                                    -------- Symbols --------      Pages       Processing
;       File                         Total    Loaded   Percent      Mapped      Time

;   _$255$DUA28:[SYSLIB]LIB.L32;1    18619        31         0        1000        00:01.8



;                                   COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/VARIANT:1/LIS=LIS$:PATSPA/OBJ=OBJ$:PATSPA MSRC$:PATSPA/UPDATE=(ENH$:PATSPA)

; Size:          999 code + 128 data bytes
; Run Time:        00:34.2
; Elapsed Time:    02:04.8
; Lines/CPU Min:    3479
; Lexemes/CPU-Min: 37166
; Memory Used:  213 pages
; Compilation Complete
```

PATREB
LIS

PATSCA
LIS

PATSIO
LIS

PATRST
LIS

PATSPA
LIS

PATSSV
LIS